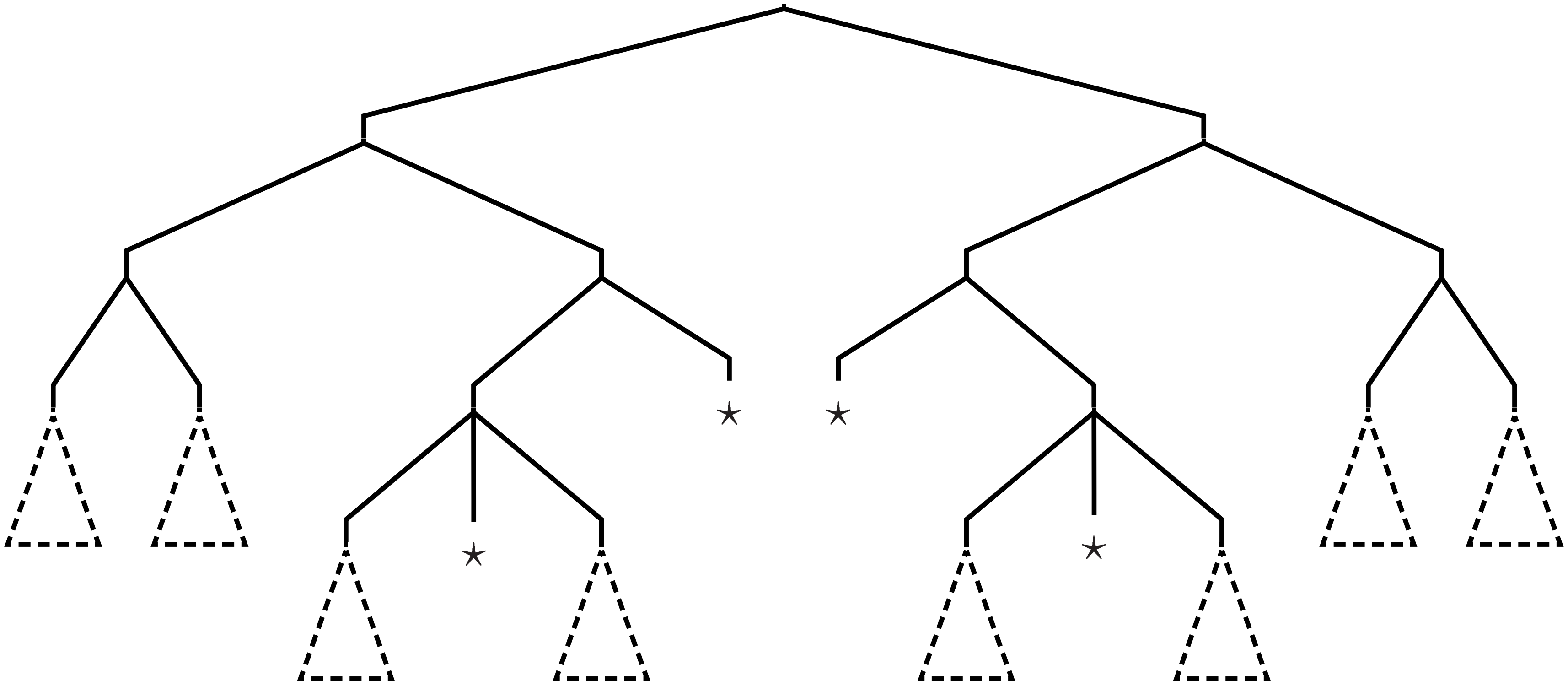
Second Part of the Tutorial

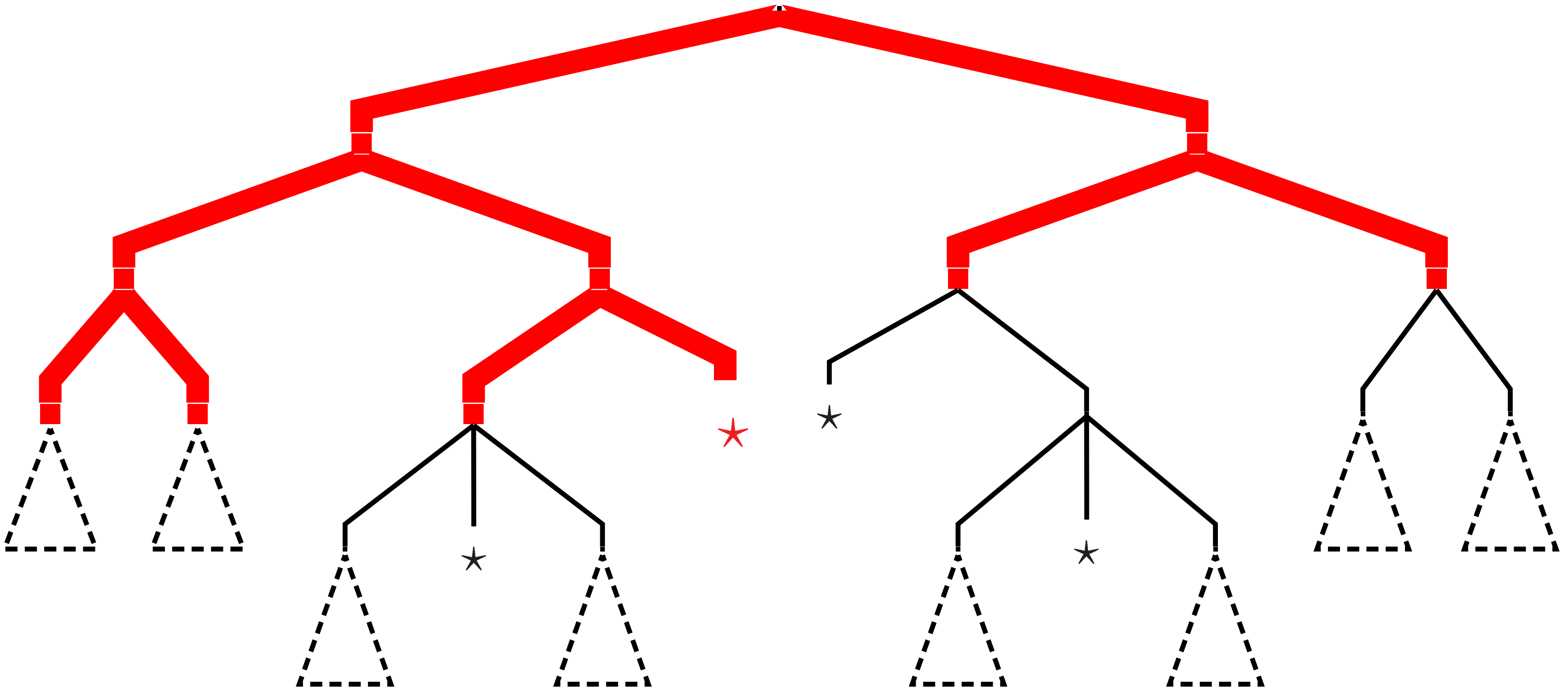
Strongly Complete Proof Procedures for Free Variable Tableaux

- Strategies for proof search:
  - breadth-search first search
  - depth-first search with iterative deepening
  - depth-first search with a strongly complete proof procedure
- Making destructive calculi “weakly” non-destructive
- Fairness for “weakly” non-destructive calculi: Making sure that
  - the proof advances
  - according to an ordering on formulae
- Example for proof construction
- Concluding remarks

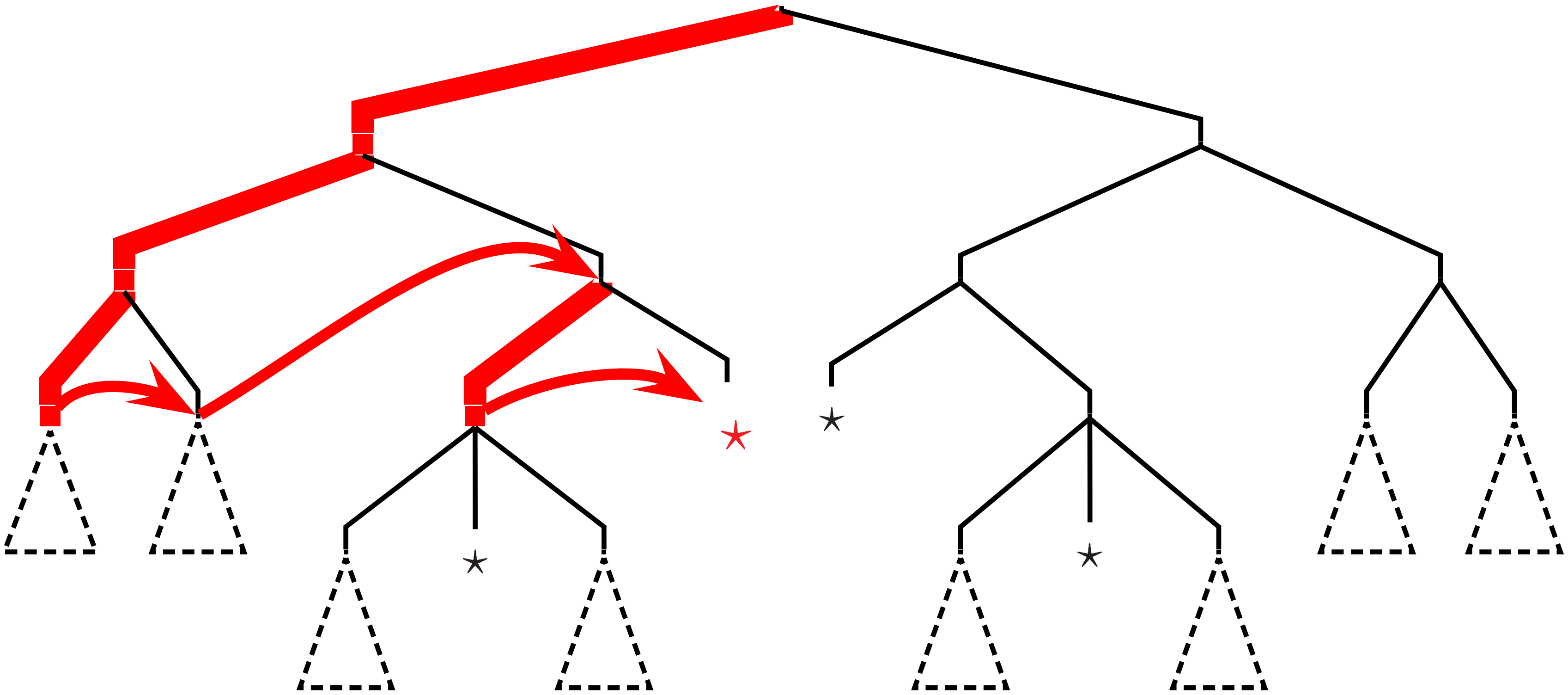
The Search Space of Proof Confluent Calculi



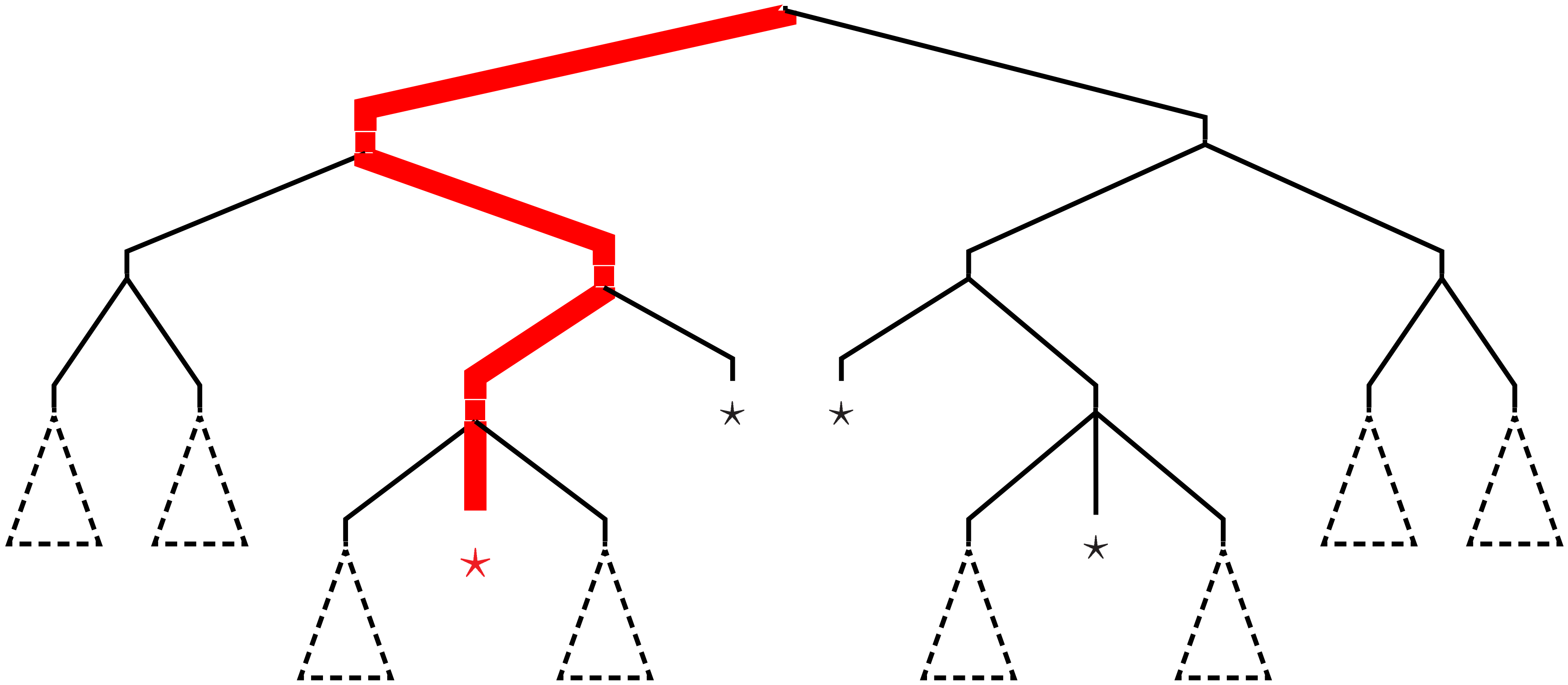
Breadth-first Search



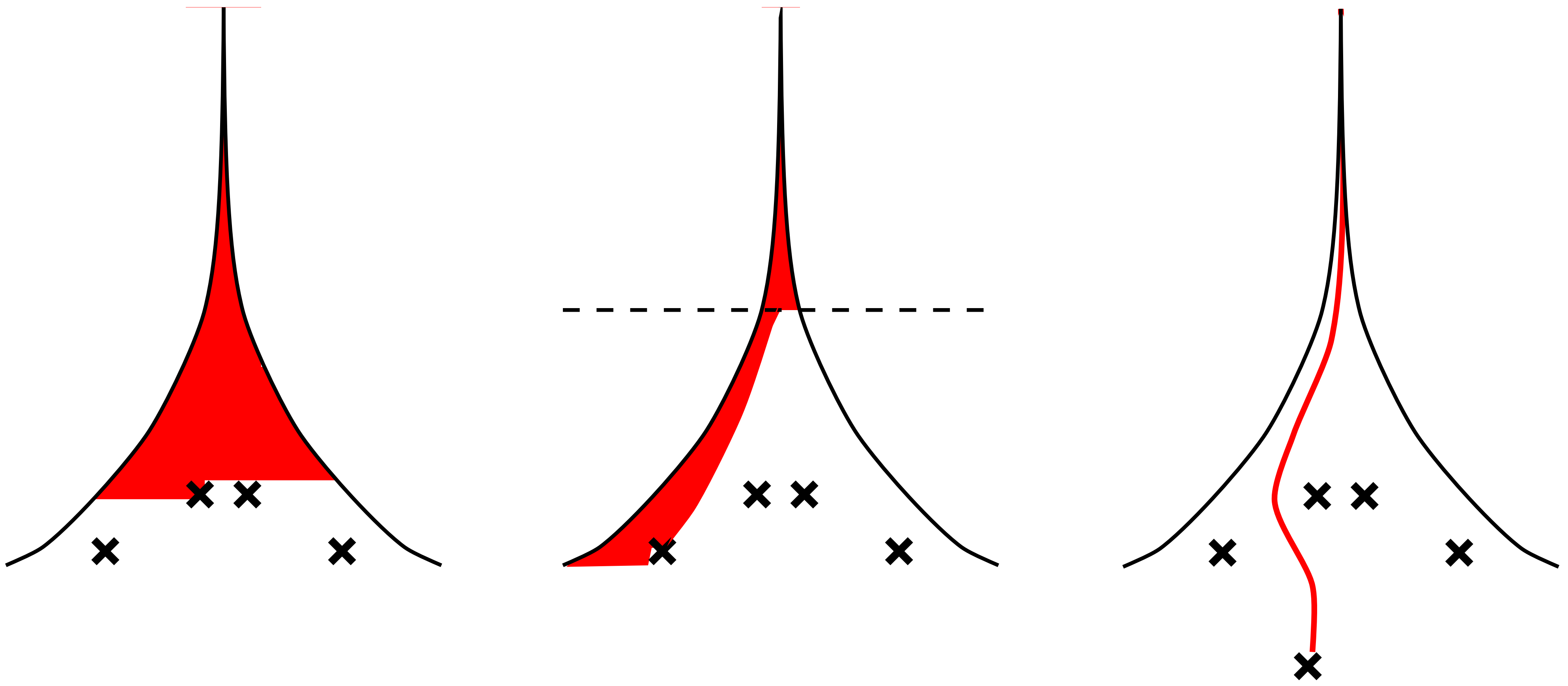
Depth-first Search with Iterative Deepening



Depth-first Search with a Strongly Complete Proof Procedure



Search Strategies: Comparison



Breadth-first search

Depth-first search with iterative deepening

Depth-first search with strongly complete proof procedure

Designing Strongly Complete Proof Procedures for Proof-confluent Destructive Calculi

The Idea:

First: Make the calculus "weakly" non-destructive.

The destroyed parts of a tableau are reconstructed immediately after they have been destroyed.

Second: Use a fairness strategy. Make sure that the proof

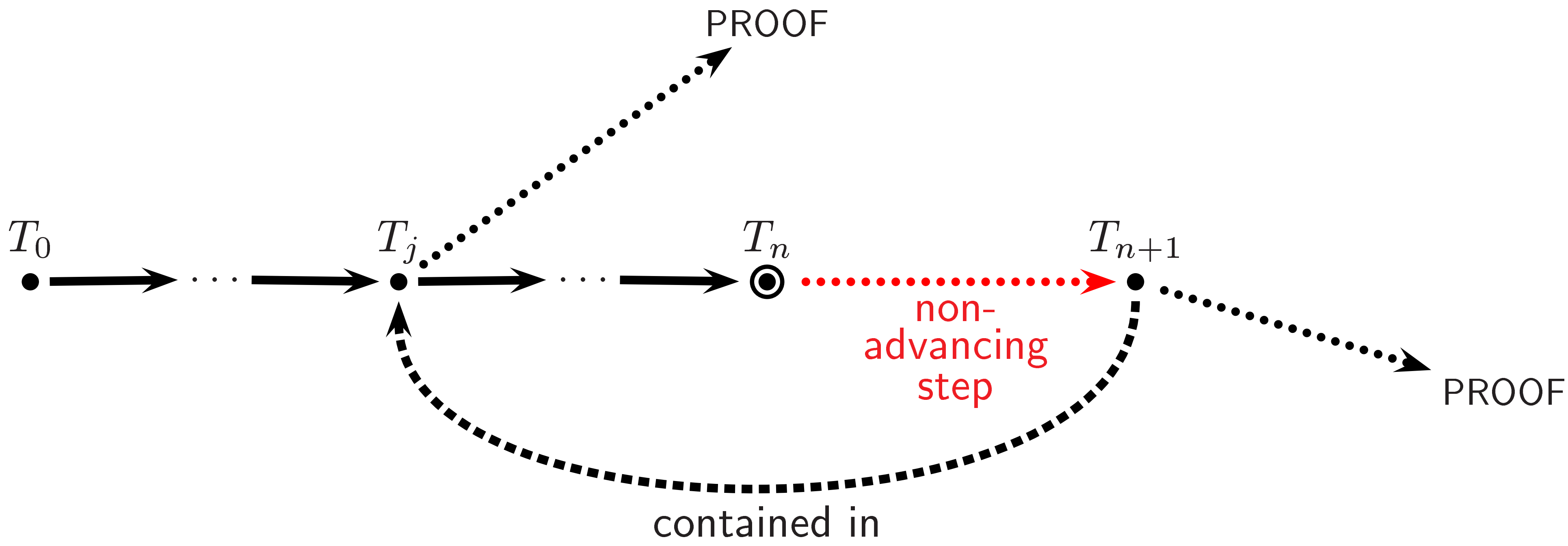
- advances at each step (no cycles)
- according to an ordering on literals

Similar approach for a connection calculus based on matrix representation:

Baumgartner, Eisinger, Furbach (1999)

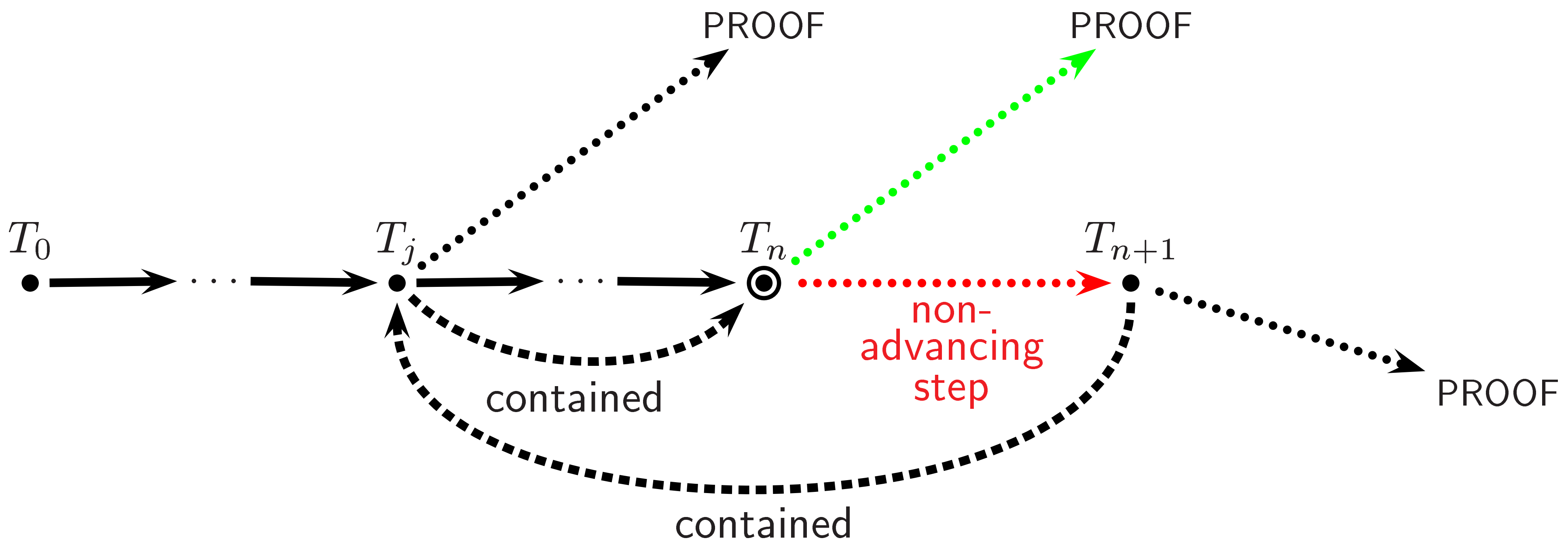
The Effect of (Non-)Destructiveness

In *destructive* calculi, a tableau *may not* be contained in its successor tableaux.



The Effect of (Non-)Destructiveness

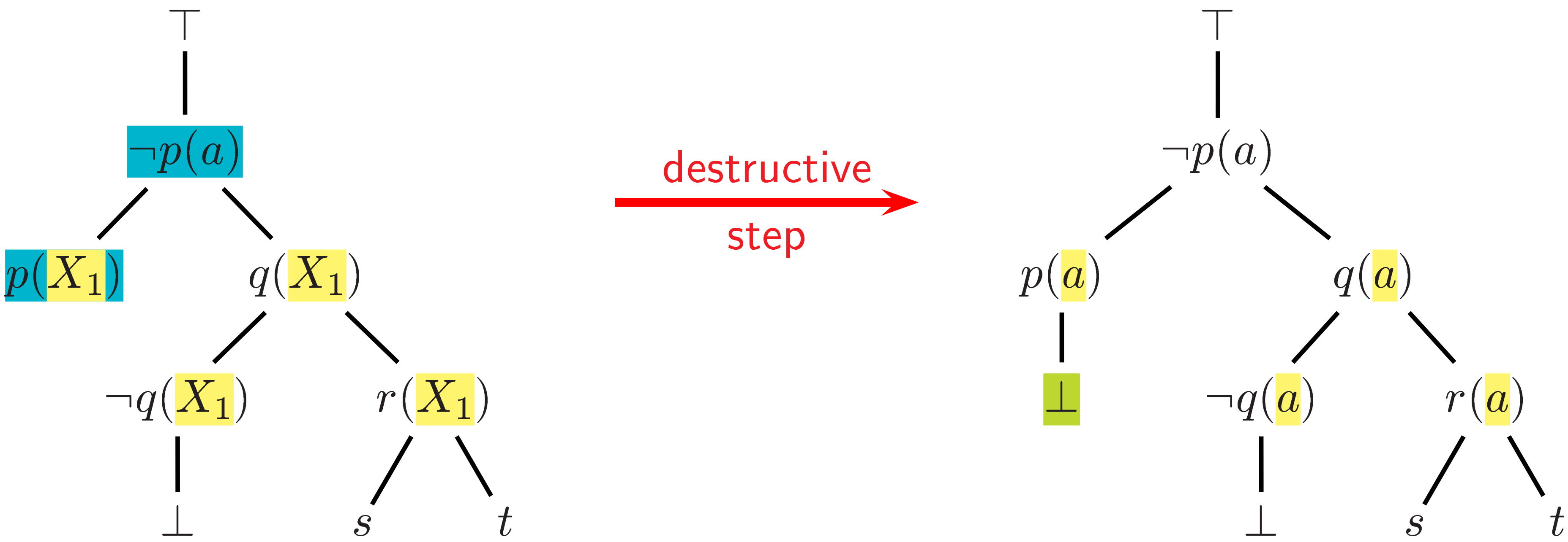
In *non-destructive* calculi, tableaux *are* contained in their successor tableaux.



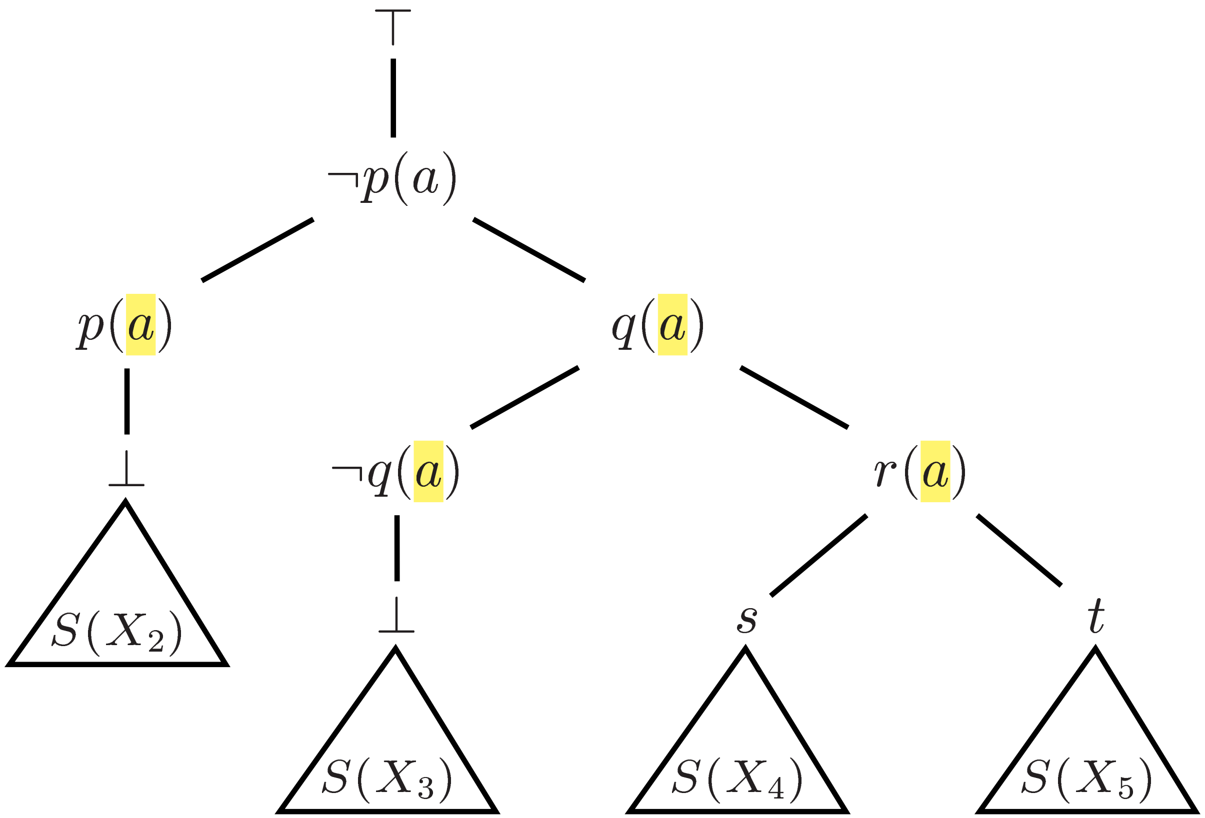
Non-destructive calculi are still proof confluent when restricted to advancing proofs steps.

Reconstruction Steps: Example (I)

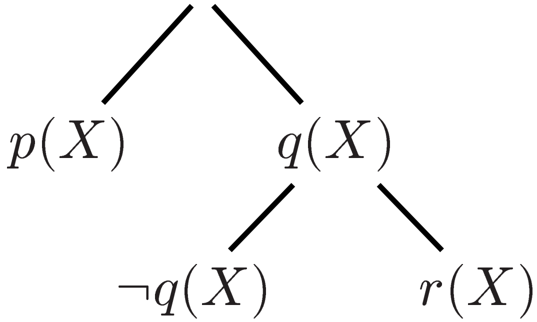
$$S = \{ \quad \neg p(a), \quad p(X) \vee q(X), \quad \neg q(X) \vee r(X), \quad s \vee t \quad \}$$



Reconstruction Steps: Example (II)



where $S(X) =$



Avoiding Proof Cycles:
A Criterion for Recognizing Non-advancing Proof Steps

$T_0, \ldots, T_n \quad \leadsto \quad T_{n+1}$ is a non-advancing deduction step

if

some T_j contains T_{n+1}

where

T_j contains T_{n+1}

if

each branch of T_j contains some branch of T_{n+1}

where

A Criterion for Recognizing Non-advancing Proof Steps (cont.)

A branch B of T contains a branch B' of T'

if

for all sets Φ' of literals on B' with at most two elements

there is a set Φ of literals on B such that

1. $\Phi\pi = \Phi'$ (π a variable renaming)

2. for each literal L in T that has variables in common with Φ there is an L' in T' such that $L\pi = L'$ up to renaming of free variables not occurring in Φ'

A Criterion for Recognizing Non-advancing Proof Steps: Intuition

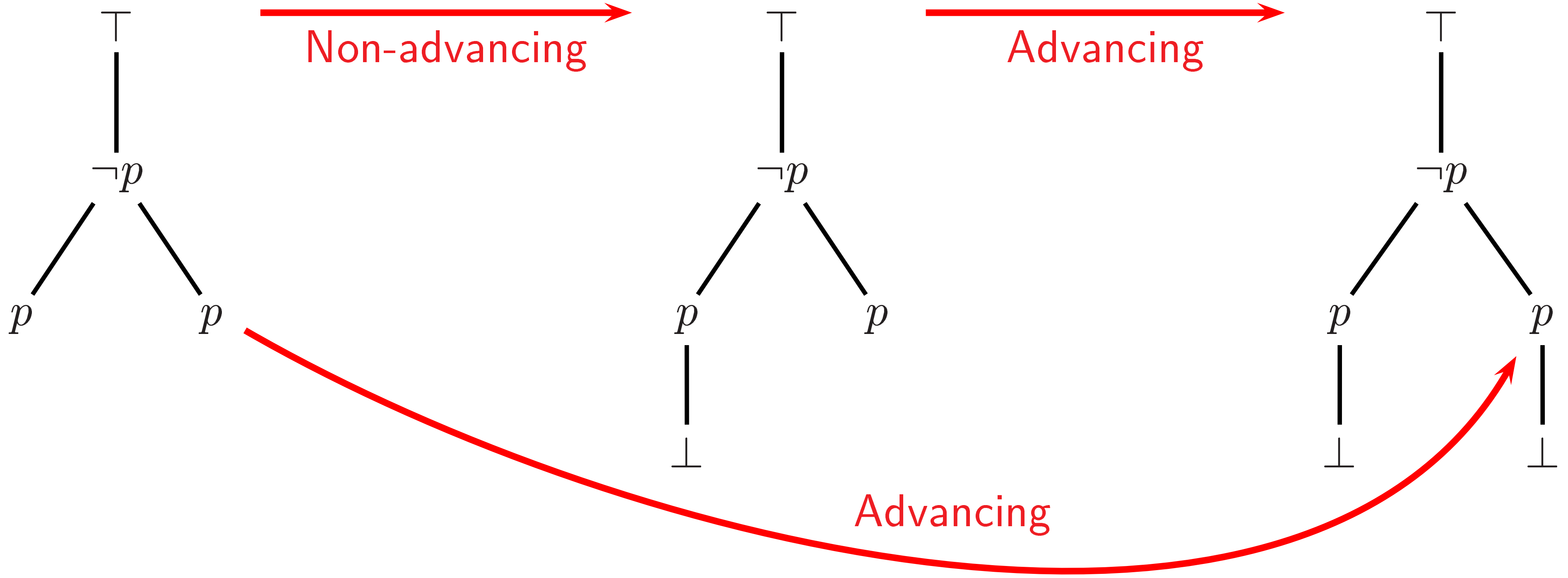
$p(a)$ allows to derive $r(a)$ (e.g. using the clause $\neg p(a) \vee r(a)$)

B contains B'

B	somewhere in T	B'	somewhere in T'
$p(X)$	$q(X, Y)$	$p(X')$	$q(X', Z)$
$p(a)$	$q(a, Y)$	$p(a)$	$q(a, Z)$
$r(a)$		$r(a)$	

The Notion of Proof Steps

The same rule application on several branches is considered to be a single proof step



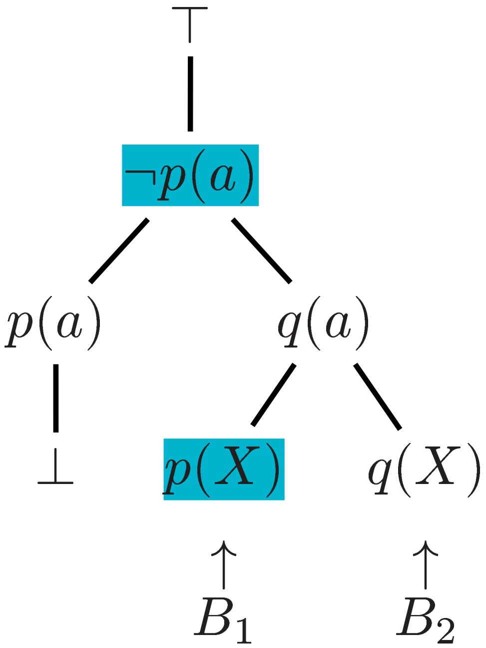
Considered to be one step

Recognizing Non-advancing Proof Steps: Example (I)

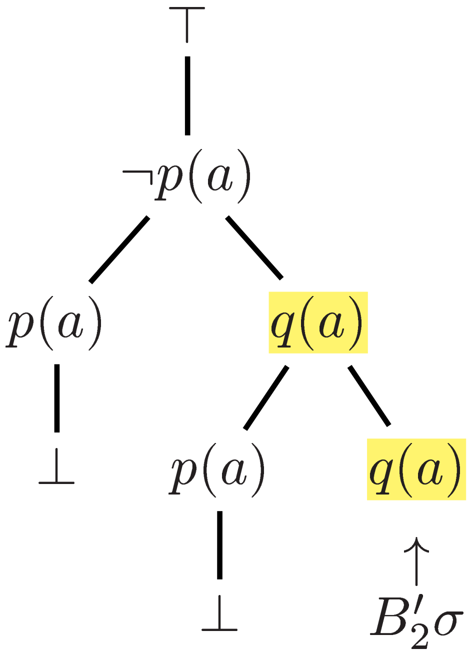
$$\begin{array}{cc} \multicolumn{2}{c}{\top} \\ / & \backslash \\ p(X) & q(Y) \end{array} \qquad \text{contains} \qquad \begin{array}{cc} \multicolumn{2}{c}{\top} \\ / & \backslash \\ p(X') & q(X') \end{array}$$

but *not* vice versa

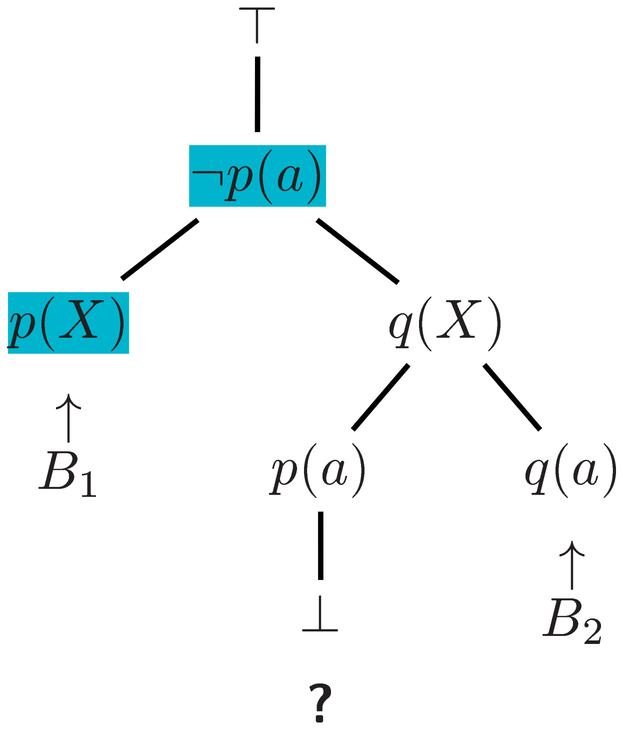
Recognizing Non-advancing Proof Steps: Example (II)



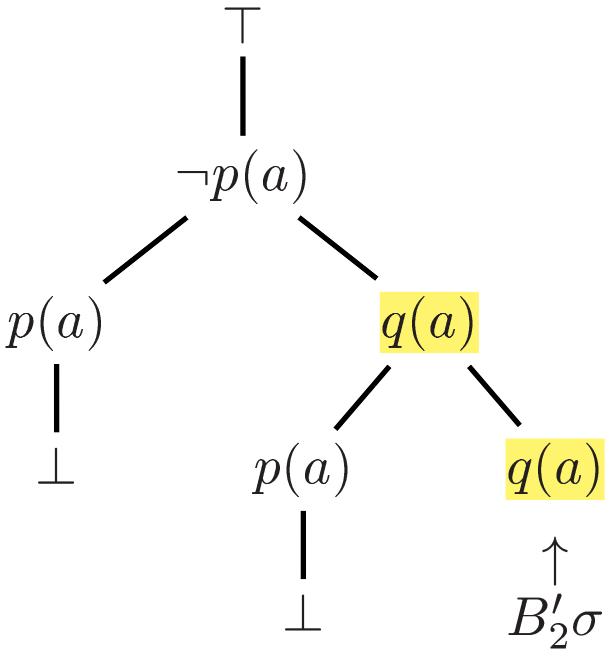
contains



Recognizing Non-advancing Proof Steps: Example (III)



does *not* contain



A Criterion for Recognizing Non-advancing Proof Steps: Theorem

If S is an unsatisfiable clause set,

then there is a closed tableau T for S

such that each step in the construction of T is advancing.

Weight Orderings

Each class of literals that are identical up to variable renaming is assigned a weight $w \in \mathbb{N}$ such that:

- only finitely many literal classes are assigned the same weight
- if L' is an instance of L, then $w(L') > w(L)$

Tableau rule applications are compared according to the maximum of the weights of all literals they add to a tableau.

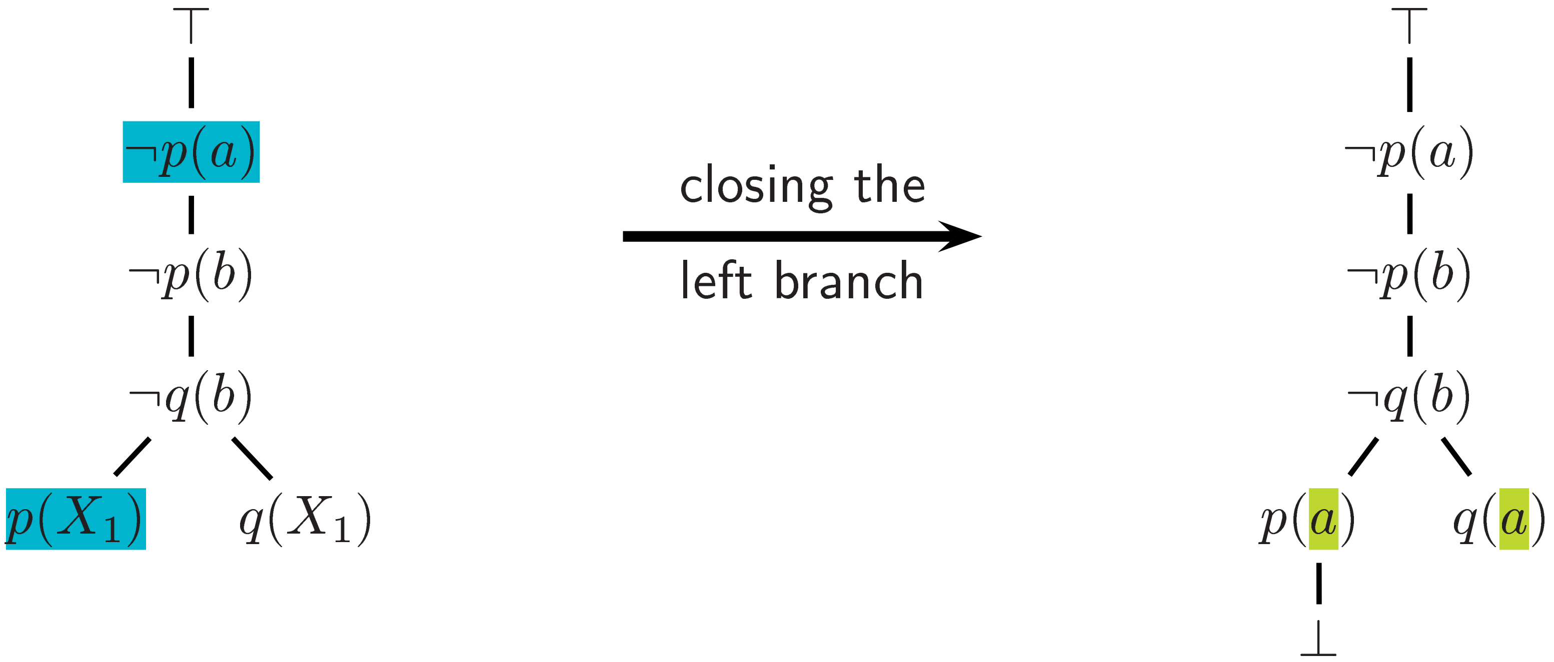
Constructing a Strongly Complete Proof Procedure: Theorem

A proof procedure for free variable clause tableaux that constructs a sequence $(T_i^+)_{i\geq 1}$ of tableaux as described below is strongly complete.

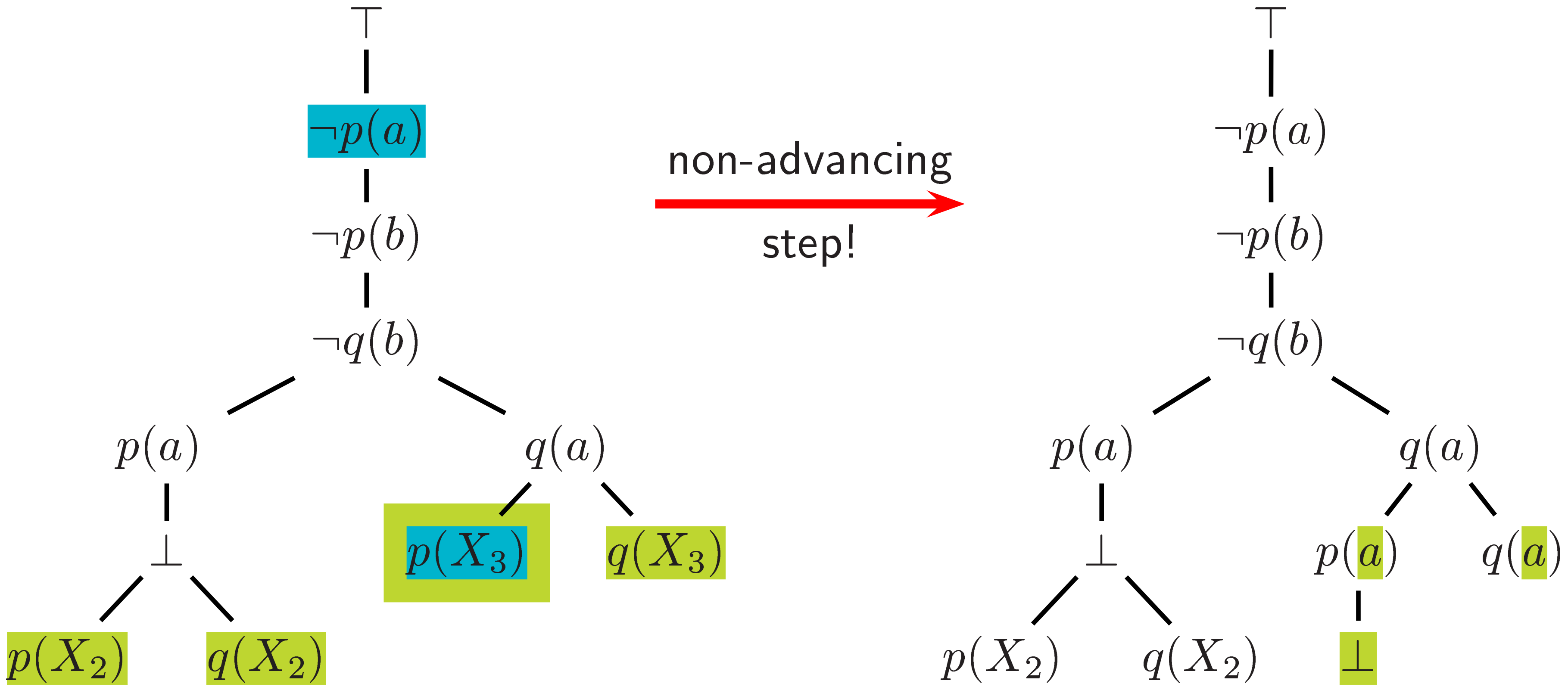
- $T_1^+ = \top$ (the initial tableau)
- T_{i+1} is a successor tableau of T_i^+ such that
  (a) T_{i+1} is not contained in T_i^+,
  (b) there is no T'_{i+1} satisfying (a) with a smaller maximal literal weight.
- T_{i+1}^+ is derived from T_{i+1} by reconstruction of the destroyed parts.

Example (I)

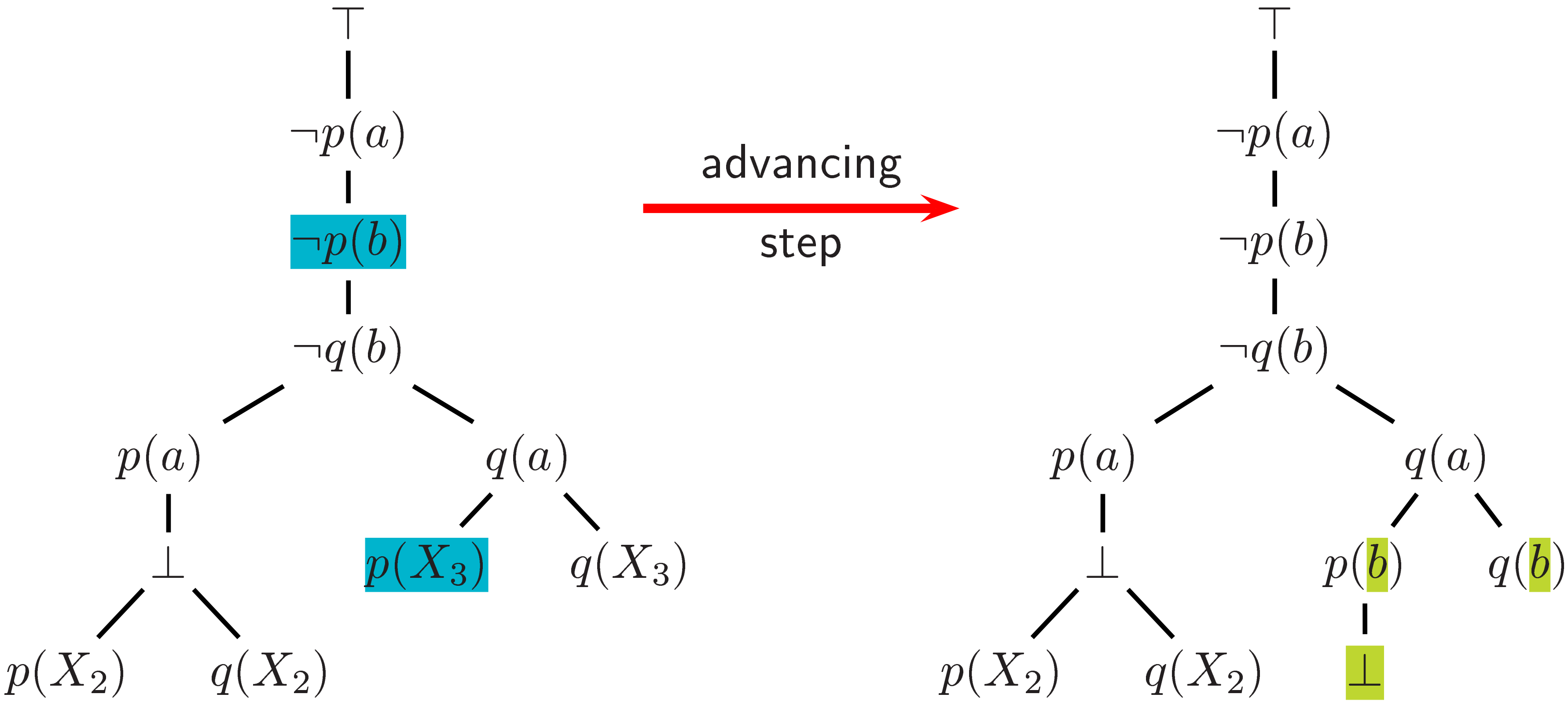
$$S = \{ \quad \neg p(a), \quad \neg p(b), \quad \neg q(b), \quad p(X) \lor q(X) \quad \}$$



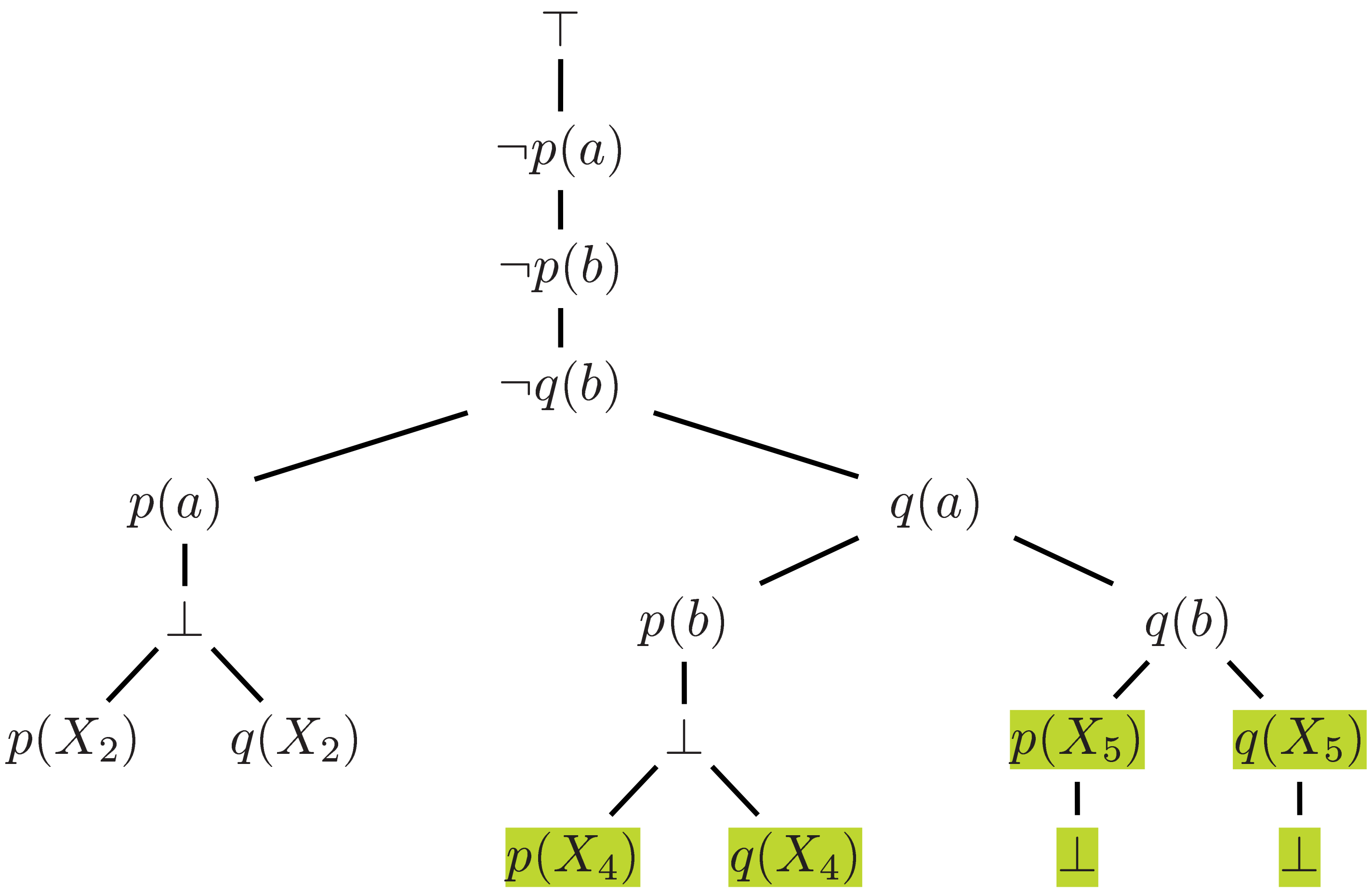
Example (II)


$\top$
$\neg p(a)$
$\neg p(b)$
$\neg q(b)$
$p(a)$
$q(a)$
$\perp$
$p(X_3)$
$q(X_3)$
$p(X_2)$
$q(X_2)$
non-advancing
step!
$\top$
$\neg p(a)$
$\neg p(b)$
$\neg q(b)$
$p(a)$
$q(a)$
$\perp$
$p(a)$
$q(a)$
$p(X_2)$
$q(X_2)$
$\perp$

Example (III)


$\top$
$\neg p(a)$
$\neg p(b)$
$\neg q(b)$
$p(a)$
$q(a)$
$\perp$
$p(X_3)$
$q(X_3)$
$p(X_2)$
$q(X_2)$
advancing
step
$\top$
$\neg p(a)$
$\neg p(b)$
$\neg q(b)$
$p(a)$
$q(a)$
$\perp$
$p(b)$
$q(b)$
$p(X_2)$
$q(X_2)$
$\perp$

Example (IV)



$\top$
$\neg p(a)$
$\neg p(b)$
$\neg q(b)$
$p(a)$
$q(a)$
$\bot$
$p(b)$
$q(b)$
$p(X_2)$
$q(X_2)$
$\bot$
$p(X_5)$
$q(X_5)$
$p(X_4)$
$q(X_4)$
$\bot$
$\bot$

Concluding Remarks

These slides, references, and related papers are available at:

`http://i12www.ira.uka.de/tab99-tutorial`